Theoretical Computer Science 459 (2012) 77–88

Contents lists available at SciVerse ScienceDirect

Theoretical Computer Science

journal homepage: www.elsevier.com/locate/tcs

Algorithms for dominating clique problems☆

N. Bourgeois [a], F. Della Croce [b], B. Escoffier [a,*], V.Th. Paschos [a,c]

[a] PSL Research University, Université Paris-Dauphine, LAMSADE, CNRS UMR 7243, France

[b] D.A.I., Politecnico di Torino, Italy

[c] Institut Universitaire de France, France

ARTICLE INFO

Article history:
Received 15 December 2011
Received in revised form 9 July 2012
Accepted 12 July 2012
Communicated by G. Ausiello

Keywords:
Dominating clique
Exponential time algorithms
Exact and approximation algorithms

ABSTRACT

We handle in this paper three dominating clique problems, namely, the decision problem to detect whether a graph has a dominating clique and two optimization versions asking to compute a maximum- and a minimum-size dominating clique of a graph G, if G has a dominating clique. For the three problems we propose exact moderately exponential algorithms with worst-case running time upper bounds improving those by Kratsch and Liedloff [D. Kratsch, M. Liedloff, An exact algorithm for the minimum dominating clique problem, Theoret. Comput. Sci. 385 (1–3) (2007) 226–240]. We then study the three problems in sparse and dense graphs also providing improved running time upper bounds. Finally, we propose some exponential time approximation algorithms for the optimization versions.

1. Introduction

Given a graph $G = (V, E)$, a *dominating clique* is a clique which is also a dominating set for G, i.e., it is a subset V' of pairwise adjacent vertices such that every vertex $v \in V \setminus V'$ has a neighbor in V'. Determining whether there exists a dominating clique or not is known to be **NP**-complete [19]. This implies that both natural optimization versions of the problem, namely, MIN DOMINATING CLIQUE and MAX DOMINATING CLIQUE, asking to compute a dominating clique of *minimum* and of *maximum size*, respectively, are **NP**-hard.

These problems make part of a large family of graph problems, called domination problems in [7], whose notorious representative is MIN INDEPENDENT DOMINATING SET. Generally, the properties of dominating sets are useful in identifying structural properties of a social network [17,18] (cited in [8]) and in computing the threshold dimension of certain classes of graphs. Although less known than MIN INDEPENDENT DOMINATING SET, dominating clique problems have been already studied, in particular in [8]. The dominating clique concept has many applications, for instance in network design where, in order to set up the communication links in a network one might want a strong central group that can communicate with each member of this group so that everyone outside the group could communicate with someone within the group. In the same spirit the dominating clique concept could be used for forest fire prevention, a group of forest fire sentries that could see various sections of a forest might also be positioned in such a way that each could see the others in order to use triangulation to locate the site of a fire [8].

Obviously, all the three versions of dominating clique problems mentioned above can be solved by enumerating all the subsets of V. So, an interesting problem is to devise algorithms able to optimally solve the three problems EXISTING

☆ Research partially supported by the French Agency for Research under the DEFIS program TODO, ANR-09-EMER-010.

* Corresponding author. Tel.: +33 144054853.
E-mail addresses: bourgeois@lamsade.dauphine.fr (N. Bourgeois), federico.dellacroce@polito.it (F. Della Croce), escoffier@lamsade.dauphine.fr (B. Escoffier), paschos@lamsade.dauphine.fr (V.Th. Paschos).

0304-3975/$ – see front matter © 2012 Elsevier B.V. All rights reserved.
doi:10.1016/j.tcs.2012.07.016

Table 1
Results of Sections 3, 4.1 and 4.2.

	Former result	Our result
EXISTING DOMINATING CLIQUE	$O^*(1.3387^{\|V\|})$ [20]	$O^*(1.2628^{\|V\|})$
- exponential space allowed	$O^*(1.3234^{\|V\|})$ [20]	$O^*(1.2453^{\|V\|})$
MAX DOMINATING CLIQUE	$O^*(1.4423^{\|V\|})$ [16]	$O^*(1.3196^{\|V\|})$
- exponential space allowed		$O^*(1.2937^{\|V\|})$
MIN DOMINATING CLIQUE	$O^*(1.3387^{\|V\|})$ [20]	$O^*(1.3248^{\|V\|})$
- exponential space allowed	$O^*(1.3234^{\|V\|})$ [20]	$O^*(1.298^{\|V\|})$

Table 2
Results of Section 5 where $\mu \in [0, 1]$ is an increasing function of $D(G)$.

	MIN DOMINATING CLIQUE	MAX DOMINATING CLIQUE
Max degree Δ	2^{Δ}	$3^{\Delta/3}$
Min degree $\delta \geqslant \|V\|/2$	$\binom{\|V\|}{\|V\|-\delta}$	$1.22^{\|V\|}\binom{\|V\|}{\|V\|-\delta}$
$D(G) \geqslant 3/4$	$\binom{\|V\|}{\sqrt{1-D(G)}\|V\|}$	$1.22^{\|V\|}\binom{\|V\|}{\sqrt{1-D(G)}\|V\|}$
$D(G) \leqslant 1/4$	$\binom{\|V\|}{\sqrt{D(G)}\|V\|}$	$(1+\mu/\sqrt{D(G)})^{\|V\|\sqrt{D(G)}}$

DOMINATING CLIQUE, MIN DOMINATING CLIQUE and MAX DOMINATING CLIQUE using time $O(2^{c|V|}p(|V|))$, where c is a constant less than 1 and p some polynomial function. Notice that, compared to the slightest improvement of c, p is non relevant. So, from now on, we use the notation $O^*(2^{c|V|})$ in order to omit polynomial factors.

Regarding EXISTING DOMINATING CLIQUE, the trivial $O^*(2^{|V|})$ bound can be easily improved to $O^*(3^{|V|/3}) = O^*(1.443^{|V|})$ by enumerating all maximal cliques and verifying each of them for being a dominating set. For this, just use a polynomial delay algorithm to generate all maximal independent sets [16] and take into account that the number of maximal (by inclusion) independent sets in a graph is at most $3^{|V|/3}$ [21]. Recently, [20] have proposed a branching algorithm that, according to a measure and conquer analysis [12], solves MIN DOMINATING CLIQUE with polynomial space and running time $O^*(1.3387^{|V|})$, and another one that requires $O^*(1.3234^{|V|})$ time and space. Naturally, these algorithms also solve EXISTING DOMINATING CLIQUE.

In what follows, we devise in Section 3 an exact algorithm solving EXISTING DOMINATING CLIQUE in $O^*(1.2628^{|V|})$ using polynomial space. Using memoization, we improve this time bound down to $O^*(1.2453^{|V|})$ but using exponential space. In Section 4.1, we settle MAX DOMINATING CLIQUE and propose an algorithm with running time $O^*(2^{2|V|/5}) = O^*(1.3196^{|V|})$ using polynomial space. Furthermore, we prove that this running time is "tight" in the sense that we exhibit an instance in which time $O^*(2^{2|V|/5})$ is also a lower bound. Allowing exponential space, we decrease the running time to $O^*(1.2937^{|V|})$. In Section 4.2 we study MIN DOMINATING CLIQUE and propose an algorithm with tight running time $O^*(1.3248^{|V|})$ using polynomial space; once more, allowing exponential space, we decrease the running time to $O^*(1.298^{|V|})$. Table 1 summarizes the results of Sections 3, 4.1 and 4.2.

In Section 5, we restrict ourselves to sparse and dense graphs, and produce exact algorithms whose complexities depend on the minimum, maximum, and average degrees of the input graph. These results are summarized in Table 2 (some of them being obvious), where $D(G) = 2|E|/(|V|(|V|-1))$ is the density of the graph G. For instance, we show in Section 5 that if $n - \delta = o(n)$ (where δ is the minimum degree of vertices in the graph), or if $D(G) = 1 - o(1)$, MIN DOMINATING CLIQUE can be solved in subexponential time and MAX DOMINATING CLIQUE using roughly the same running time as MAX CLIQUE. On the other hand, if $D(G) = o(1)$ both MIN and MAX DOMINATING CLIQUES can be also solved in subexponential time.

It is easy to see that no polynomial time approximation algorithm can exist for MIN DOMINATING CLIQUE and MAX DOMINATING CLIQUE, unless $\mathbf{P} = \mathbf{NP}$, since any such algorithm would first solve EXISTING DOMINATING CLIQUE that is **NP**-complete. For this reason, in Section 6, we present some approximation results for MIN DOMINATING CLIQUE and MAX DOMINATING CLIQUE by "low complexity" exponential time algorithms. The basic motivation for the development of such algorithms is to overcome the inapproximability barriers of polynomial approximation algorithms, by devising approximation strategies that, though with exponential running time, run much faster than the corresponding exact algorithms known for these problems. The interested reader can find more about this issue and its motivations in [4], while several results for well-known problems can also be found, for instance, in [1–3,9,10,13,22]. Recall that the approximation ratio of an algorithm A supposed to solve an **NP**-hard problem Π is defined by $\max\{m(I,S)/\operatorname{opt}(I), \operatorname{opt}(I)/m(I,S)\}$ where, for any instance I of Π, $m(I,S)$ is the value of the solution S computed by A in I and $\operatorname{opt}(I)$ is the value of an optimal solution of I.

2. Preliminaries

In what follows, given a graph $G = (V, E)$ and a vertex $v \in V$, the neighborhood $N(v)$ of v is the set of vertices that are adjacent to v and the set $N[v] = N(v) \cup \{v\}$ is called the closed neighborhood of v. For the degree of v, we use the notation $d(v) = |N(v)|$; the anti-degree of v is the quantity $\bar{d}(v) = |V \setminus N[v]|$. Vertices in $V \setminus N[v]$ are called anti-neighbors of v. For any set $V' \subset V$, we write $N_{V'}(v) = N(v) \cap V'$, $N_{V'}[v] = N[v] \cap V'$, $d_{V'}(v) = |N_{V'}(v)|$ and $\bar{d}_{V'}(v) = |V' \setminus N_{V'}[v]|$; $G[V']$ is the subgraph induced by V' in G.

For simplicity, we set $n = |V|$ and $m = |E|$. For any function $\mathcal{T}$, $T(n)$ stands for the maximum running time the algorithm requires to compute $\mathcal{T}$ on a graph containing at most n vertices.

Also, let us note that we use the same notations as in [20]. More precisely, S is the set of vertices we have added to the solution under construction, D is the set of vertices we have discarded, $A = \bigcap_{s\in S} N(s) \setminus D$ is the set of vertices still available and $F = V \setminus (\bigcup_{s\in S} N[s])$ is the set of vertices that still remain to be dominated; $T(S, D, A, F)$ is a boolean function that returns TRUE if and only if there exists a dominating clique C in G such that $S \subseteq C \subseteq A \cup S$. Let us note that, once the first vertex is picked, we ensure that A and F are disjoint. As pointed out in [20], it is equivalent to solve EXISTING DOMINATING CLIQUE or to determine if there exists $v \in V$ such that $T(\{v\}, \emptyset, N(v), V \setminus N[v]) =$ TRUE. For simplicity, in the proofs that follow in Sections 3 and 4, we denote by n the number of non fixed vertices, i.e., $n = |A| + |F|$. Parameter n measures the progress of the algorithm.

We now give some immediate lemmata, that have already been stated, for instance, in [20], but that one should keep in mind in order to understand how our algorithms work.

Lemma 1. *If there exists some dominating clique K^*, then any clique that contains K^* is a dominating clique. Also, for any $v \in V$, any dominating clique containing v contains only vertices from $N[v]$.*

Lemma 2. *For any $v \in V$, if there exists some dominating clique K containing v, then there exists a dominating clique $K' \subseteq K$ with $|K'| \leqslant |V \setminus N(v)|$.*

Proof. The claimed clique K' could be, for instance, obtained by simply taking v and one neighbor in K for any vertex in $V \setminus N[v]$. □

3. EXISTING DOMINATING CLIQUE

We devise in this section an algorithm, called EDC, that solves EXISTING DOMINATING CLIQUE with running time $O^*(1.2628^n)$. Below, we first sketch Algorithm EDC. Cases 1 to 5 are simple reduction rules that do not need any branching. On the other hand, cases 6 and 7 imply various branching rules (according to the subcases examined) that, for readability, will be detailed in the proof of Proposition 1 that follows.

Let $\alpha = \min\{d_A(u), u \in F\}$ and $\beta = \min\{\bar{d}_A(v), v \in A\}$. Algorithm EDC works as follows:

1. if $F = \emptyset$, then $T(S, D, A, F) =$ TRUE;
2. else, if $\exists u \in F$, such that $d_A(u) = 0$, then $T(S, D, A, F) =$ FALSE;
3. else, if $\exists u \in A$, such that $d_F(u) = 0$, then $T(S, D, A, F) = T(S, D \cup \{u\}, A \setminus \{u\}, F)$;
4. else, if $\exists u \in A$, such that $\bar{d}_A(u) = 0$, then $T(S, D, A, F) = T(S \cup \{u\}, D \cup N_F(u), A \setminus \{u\}, F \setminus N_F(u))$;
5. else, if $\exists i \in F$, such that $d_A(i) = 1$ with $j \in N_A(i)$, then $T(S, D, A, F) = T(S \cup \{j\}, D \cup \{i\}, A \setminus \{j\}, F \setminus \{i\})$;
6. else, if $\alpha \leqslant \beta + 1$, branch on a vertex $u \in F$ (by putting in S at least one among its neighbors in A) with $d_A(u) = \alpha$ according to the following exhaustive cases:
   (a) $d_A(u) = \alpha = 2$ with $j, k \in N_A(u)$:
      i. $\bar{d}_A(j) + d_F(j) \geqslant 3$ and $\bar{d}_A(k) + d_F(k) \geqslant 3$;
      ii. $\bar{d}_A(j) = 1, d_F(j) = 1$ and $\bar{d}_A(k) + d_F(k) \geqslant 5$;
      iii. $\bar{d}_A(j) = 1, d_F(j) = 1, \bar{d}_A(k) + d_F(k) = 4$ and $j \notin N_A(k)$;
      iv. $\bar{d}_A(j) = 1, d_F(j) = 1, \bar{d}_A(k) + d_F(k) = 4$ and $j \in N_A(k)$;
      v. $\bar{d}_A(j) = 1, d_F(j) = 1, \bar{d}_A(k) + d_F(k) = 3$ and $j \notin N_A(k)$;
      vi. $\bar{d}_A(j) = 1, d_F(j) = 1, \bar{d}_A(k) + d_F(k) = 3$ and $j \in N_A(k)$;
      vii. $\bar{d}_A(j) = 1, d_F(j) = 1, \bar{d}_A(k) = 1, d_F(k) = 1$;
   (b) $d_A(u) = \alpha \geqslant 3$ and there is one vertex in $j \in N_A(u)$ such that $d_F(j) \geqslant 2$ or $\bar{d}_A(j) \geqslant \alpha$;
   (c) $d_A(u) = \alpha \geqslant 3$ and for all $j \in N(u)$ both $d_F(u) = 1$ and $\bar{d}_A(j) = \alpha - 1$;
7. else, $\alpha \geqslant \beta + 2$. Then branch on a vertex $v \in A$ (by putting in S at least one of the vertices in $A \setminus N_A(v)$) such that $\bar{d}_A(v) = \beta$ according to the following subcases:
   (a) $\bar{d}_A(v) = \beta = 1$;
   (b) $\bar{d}_A(v) = \beta \geqslant 2$.

Proposition 1. *Algorithm EDC decides whether there exists a dominating clique, or not, with running time $O^*(1.2628^n)$ and using polynomial space. If exponential space is allowed, using memoization the time needed is $O^*(1.246^n)$.*

Proof. The correctness of the algorithm easily follows from the following remarks.

- If a vertex $u \in A$ has no neighbor in F, then u is useless and can be discarded (case 3).
- For a vertex $v \in A$, if all the vertices in $A \setminus N_A[v]$ have been discarded (or in particular if $\bar{d}_A(v) = 0$) then v can be added to S (cases 4 and 7).
- At least one of the neighbors in A of any vertex in F has to be added to S (cases 5 and 6).

For the complexity of the algorithm, we consider all subcases of cases 6 and 7. Let us remark that in cases 6 and 7, $\forall i \in F$, $d_A(i) \geqslant 2$ and $\forall u \in A$, $\bar{d}_A(u) \geqslant 1$.

Case 6a Either j or k must be selected as they are the only available vertices adjacent to u. The following subcases must be considered.

- For subcase 6(a)i, if j is selected all available vertices not in $N_A(j)$ are discarded and all free vertices in $N_F(j)$ are dominated. Hence, as $\bar{d}_A(j)+d_F(j) \geqslant 3$, at least 4 vertices including j are removed. Alternatively k is selected and similarly at least 4 vertices including k are removed. Putting all the above together, we have $T(n) \leqslant 2T(n-4)$ inducing as complexity $O^*(1.1893^n)$.
- For subcase 6(a)ii, if j is selected, as $\bar{d}_A(j)+d_F(j)=2$ at least 3 vertices including j are removed. Alternatively, k is selected and similarly, as $\bar{d}_A(k)+d_F(k) \geqslant 5$, at least 6 vertices including k are removed. Putting all the above together, we have $T(n) \leqslant T(n-3)+T(n-6)$ inducing as complexity $O^*(1.174^n)$.
- For subcase 6(a)iii, we have $\bar{d}_A(k)+d_F(k)=4$. But then, if $d_F(k)=1$, k can be discarded and j can be selected as they are both adjacent to $u \in F$ with $d_F(j)=d_F(k)=1$ but $\bar{d}_A(k)=3 > \bar{d}_A(j)=1$ with $k \notin N_A(j)$. Similarly, if $d_F(k)=3$, j can be discarded and k be selected. Alternatively, $\bar{d}_A(k)=d_F(k)=2$ and let l be the available vertex not in $N(k)$ with l adjacent to at least one vertex α in F. Then, a branch on l can be executed. If l is selected, then α is dominated, k is discarded and, correspondingly (now $\bar{d}_A(j)=0$), j is selected and u is dominated. Besides, if l is discarded, $\bar{d}_A(k)=1$ and $d_F(k)=2$; hence, j can be discarded, k selected and u dominated. Overall, we have $T(n) \leqslant 2T(n-5)$ inducing as complexity $O^*(1.1487^n)$.
- For subcase 6(a)iv, if j is selected, as $\bar{d}_A(j)+d_F(j)=2$ at least 3 vertices including j are removed. Alternatively, j is discarded, k is selected and similarly, as $\bar{d}_A(k)+d_F(k)=4$ and $j \in N(k)$, at least 6 vertices including j and k are removed. Putting all the above together, we have $T(n) \leqslant T(n-3)+T(n-6)$ inducing as complexity $O^*(1.174^n)$.
- For subcase 6(a)v, we have $\bar{d}_A(k)+d_F(k)=3$ with $k \notin N_A(j)$. But in this case, if $d_F(k)=1$, k can be discarded and j can be selected as they are both adjacent to $u \in F$ with $d_F(j)=d_F(k)=1$ but $\bar{d}_A(k)=2 > \bar{d}_A(j)=1$ and $k \notin N_A(j)$. Similarly, if $d_F(k)=2$, j can be discarded and k selected. No branch occurs for this subcase.
- For subcase 6(a)vi, let, w.l.o.g., $l \in A$, $l \notin N(j)$. If k is selected, as $\bar{d}_A(k)+d_F(k)=3$, three other vertices are removed including u. But then, j can be discarded (u being already dominated by k). Alternatively k is discarded, j is selected, l is discarded and u is dominated. Putting all the above together, we have $T(n) \leqslant T(n-4)+T(n-5)$ inducing as complexity $O^*(1.1674^n)$.
- For subcase 6(a)vii, if j and k are non adjacent, then no branch occurs as one can arbitrarily select j and discard k or vice versa. Alternatively $j \in N(k)$. Let, w.l.o.g., $l \in A$, $l \notin N(j)$ and $m \in A$, $m \notin N(j)$. If $l=m$, again then no branch occurs as one can arbitrarily select j and discard k or vice versa. Finally, if $l \neq m$, let, w.l.o.g., $\alpha \in F$, $\alpha \in N(l)$. A branch on l can be executed. If l is selected, then α is dominated, j must be discarded and, correspondingly (now $\bar{d}_A(k)=0$), k is selected, u is dominated and m is discarded. Alternatively, l is discarded and correspondingly (now $\bar{d}_A(j)=0$) j is selected, u is dominated and k is discarded (now $d_F(k)=0$). Hence, we have $T(n) \leqslant T(n-4)+T(n-6)$ inducing as complexity $O^*(1.151^n)$.

In all, for case 6a, the worst-case recursion is $T(n) \leqslant 2T(n-4)$; so, $T(n)=O^*(1.1893^n)$.

Case 6b We prove by recurrence on α that the worst case leads to $T(n) \leqslant \alpha T(n-\alpha-2)$, which is the previous result for $\alpha=2$. In the considered case, either we select j or we discard it. Selecting j allows to remove $d_F(j)+\bar{d}_A(j)+1 \geqslant 2+\alpha$ vertices. If we do not select j, we remove one vertex, and get an instance G' where $\alpha' = \alpha-1$ and $\beta' \geqslant \beta-1$, so $\alpha' \leqslant \beta'+1$ hence at worst by recurrence we have G' $(\alpha-1)T(n'-(\alpha+1)) = (\alpha-1)T(n-(\alpha+2))$ subinstances from the branching in G'. In all, α branches where we remove $\alpha+2$ vertices with worst-case for $\alpha=4$ corresponding to $T(n) \leqslant 4T(n-6)$ with complexity $O^*(1.26^n)$.

Case 6c Suppose that there is an edge (j_1, j_2) in $N(u)$. When selecting j_1, we can assume that we do not select the other neighbors of u, since $d_F(j_1)=1$. Then, selecting j_1 allows to remove $\bar{d}_A(j_1)+3 \geqslant \alpha+2$ vertices (the anti-neighbors of j_1, as well as j_1, j_2 and u). As previously, not selecting j_1 leads to $(\alpha-1)$ branches where $\alpha+2$ vertices are removed. Suppose now that $N(u)$ is an independent set. Since for any $j \in N(u)$ both $d_F(u)=1$ and $\bar{d}_A(j)=\alpha-1$, all vertices in $N(u)$ are equivalent (same neighborhood), hence we select one of them at random without branching.

Case 7a Let w be the anti-neighbor of v in A. The branching is either to take w (and to discard v), or to discard w and take v ($N[v]=A$ after discarding w). If both $d_F(v)+\bar{d}_A(v) \geqslant 3$ and $d_F(w)+\bar{d}_A(w) \geqslant 3$, then we get two branches of size at most $n-4$. Assume now that one vertex, say v, verifies $d_F(v)=\bar{d}_A(v)=1$ (permute v and w otherwise). Let u be the unique neighbor of v in F. Then, if we take v we can discard all the other neighbors of u in A (since taking one of these vertices would allow to remove v from the clique). Since $\alpha \geqslant \beta+2=3$, in the branch we take v, we remove at least 4 vertices. If $d_F(w)+\bar{d}_A(w) \geqslant 3$, in the other branch (taking w) at least 4 vertices are also removed. The only remaining case is when $d_F(v)=\bar{d}_A(v)=d_F(w)=\bar{d}_A(w)=1$. If u is also a neighbor of w then we can remove w and take v (or vice versa) without branching. Otherwise, when taking v we remove at least 5 vertices (v, u, two other neighbors of u in A, and w), while we remove at least 3 vertices when taking w. Overall, the worst-case is $T(n)=T(n-3)+T(n-5)$ with complexity $O^*(1.1939^n)$ which corresponds to $T(n) \leqslant (\beta-1)T(n-\beta-3)+T(n-\beta-2)+T(n-\beta-4)$ for $\beta=1$.

Case 7b We show the previous inequality by recurrence on β. Let w be some anti-neighbor of v. We branch on w. Suppose that we select it. If $d_F(w) \geqslant 2$, then we remove $\beta+3$ vertices. Otherwise, when selecting w we can remove the neighbors of the only vertex dominated by w in F, hence in all $\alpha+1 \geqslant \beta+3$ vertices. On the other hand, if we do not select w, we get an instance G' with $\alpha' \geqslant \alpha-1$ and $\beta'=\beta-1$ with again $\alpha' \geqslant \beta'+2$. Thus, for $\beta=2$, either we

select w and remove $\beta+3=5$ vertices, or we remove w and generate a subproblem with $\beta=1$ and worst-case complexity $T(n')\leqslant T(n'-3)+T(n'-5)$ for $n'=n-1$. Overall for $\beta=2$, we get $T(n)\leqslant T(n-4)+T(n-5)+T(n-6)$. By recurrence, this induces $T(n)\leqslant(\beta-1)T(n-\beta-3)+T(n-\beta-2)+T(n-\beta-4)$ for general β the worst-case being reached for $\beta=3$ where we have $T(n)\leqslant 2T(n-6)+T(n-5)+T(n-7)$ with complexity $O^*(1.2628^n)$.

The overall worst-case complexity is then $O^*(1.2628^n)$.

We now show how the time bound claimed in Proposition 1 can be further improved by memoization. Obviously, this improvement requires exponential space. The principle of memoization, as it has been explained for example in [23,12], is quite simple. Before running the algorithm on the main graph, we run it on every induced subgraph $G[X]$ of size at most αn, and for each X and each admissible partition (A,F) of X. Note that there exists a linear number of such partitions since $A=N_X(v)$, for $v\in V\setminus X$ (and $F=X\setminus A$). We store the results in a table. [1] Notice that, thanks to the recurrence defined just above, we only need to call a finite number of subproblems of size $k-1$ or less in order to compute a given subproblem of size k. Then, using a classical bottom up technique where we solve the problem on subinstances of increasing size, the total computation time (and space) for all the subproblems of size k, say $S(k,G)$, is at most:

$$O^*\left(\binom{n}{k}+\sum_{i\leqslant k-1}S(i,G)\right)$$

from which we get (for $\alpha\leqslant 1/2$):

$$\sum_{k\leqslant\alpha n}S(k,G)\in O^*\left(\binom{n}{\alpha n}\right).$$

We run the main algorithm until the remaining graph has size at most αn; then a polynomial-time query in the storage table allows us to conclude. Thus, the total running time and space is:

$$O^*\left(\max\left\{\binom{n}{\alpha n},\,1.2628^{(1-\alpha)n}\right\}\right).$$

This value is minimum for an α solution of the equation $1.2628^{1-\alpha}=1/(\alpha^\alpha(1-\alpha)^{1-\alpha})$ (using Stirling's formula), leading to $T(n)=O^*(1.246^n)$ and completing the proof of the proposition. □

4. MAX and MIN DOMINATING CLIQUES

4.1. MAX DOMINATING CLIQUE

The function $T(S,D,A,F)$ is now an integer function that returns the cardinality of a maximum dominating clique in G contained in $S\cup A$ and containing S and with no vertex from set D, if any, and $-\infty$ otherwise. Once again, solving MAX DOMINATING CLIQUE is equivalent to finding $\max_{v\in V}\{T(\{v\},\emptyset,N(v),V\setminus N[v])\}$. Note that now we cannot discard as in Algorithm EDC any vertex that has degree 0 in F, because we could be led to a solution that is not a maximum one; however, we can compute a solution by the following algorithm called MDC:

1. if $A=F=\emptyset$, then $T(S,D,A,F)=|S|$;
2. if $\exists u\in F$, such that $d_A(u)=0$, then $T(S,D,A,F)=-\infty$;
3. fix $w\in A$ such that $d_A(w)$ is maximum, and $d_F(w)$ is maximum among vertices of maximum d_A. If there exists $u\in A\setminus N[w]$, such that $d_F(u)=0$, then:

$$T(S,D,A,F)=\max_{u'\in A\setminus(N_A(w)\cup\{u\})}\left\{T\left(S\cup\left\{u'\right\},D\cup\left(A\setminus N_A\left[u'\right]\right)\cup N_F\left(u'\right),N_A\left(u'\right),F\setminus N_F\left(u'\right)\right)\right\}$$

otherwise:

$$T(S,D,A,F)=\max_{u\in A\setminus N_A(w)}\left\{T\left(S\cup\{u\},D\cup(A\setminus N_A[u])\cup N_F(u),N_A(u),F\setminus N_F(u)\right)\right\}.$$

As for EDC, if every vertex in $A\setminus N[w]$ has been discarded, then any vertex still available is in $N_A[w]$, so we can safely add w to S; here, correctness of the algorithm follows from the fact that if every vertex in $A\setminus N(w)$ but u and w have been discarded, then only one among u and w may be added; furthermore, if $d_F(u)=0$, we can safely discard u and add w to S.

Proposition 2. *Algorithm MDC computes a maximum-size dominating clique, if any, with running time $O^*(2^{2n/5})=O^*(1.3196^n)$ and using polynomial space. This bound is tight. If exponential space is allowed then, using memoization, Algorithm MDC computes a maximum-size dominating clique, if any, with running time and space $O^*(1.2937^n)$.*

Proof. The proof of the first time-bound (case of polynomial space) claimed in Proposition 2 is a straightforward consequence of the following observation:

if $\bar{d}_A(w)=\delta$, for some $\delta\in\mathbb{N}$, then:

$$T(n)\leqslant\max\left\{\delta\cdot T(n-\delta-1),(\delta+1)\cdot T(n-\delta-2),T(n-\delta-1)+\delta\cdot T(n-\delta-3)\right\}.$$

[1] Note that, in the algorithms, available vertices never become free (or *vice versa*), hence the number of subproblems of size αn to consider is $\binom{n}{\alpha n}$.

Indeed, let $(u_i)_{i\leqslant\delta}$ be the δ anti-neighbors of u. If one of these δ vertices, say u_1, is such that $d_F(u_1) = 0$, then $(u_2, \ldots, u_\delta \in D) \Rightarrow w \in S$; so, $T(n) \leqslant \delta T(n-\delta-1)$. Otherwise, $\forall i \leqslant \delta$, $d_F(u_i) \geqslant 1$. In this case, if for say u_1 $\bar{d}_A(u_1) = \delta$ then, by the fact that $d_F(w)$ is maximum (Step 3 of Algorithm MDC), $d_F(w) \geqslant 1$, and $T(n) \leqslant (\delta+1)T(n-\delta-2)$. Finally, in any other case, for any i, $\bar{d}_A(u_i) + d_F(u_i) \geqslant \delta + 2$. Thus, $T(n) \leqslant T(n-\delta-1) + \delta T(n-\delta-3)$.

Then, in order to conclude, it suffices to observe that $T(n) = 2^{2n/5}$ verifies all recurrences stated in the quoted observation just above.

Tightness can be shown in the following graph. Consider a collection G_p of p stars of size 5 (1 center plus 4 outer vertices). Set A is the set of the outer vertices, and set F is the set of the centers. Any outer vertex u is adjacent to any other outer vertex in any other star; so, $d_A(u) = 4p-4$. When the algorithm branches on any of these outer vertices, it has to solve EXISTING DOMINATING CLIQUE on four identical copies of G_{p-1} and, in this case $T(n) = \Omega(4^{n/5})$.

We now show the second time-bound claimed (case of exponential space). As previously, we run MDC on every subgraph of size at most βn and store the results in a table, with time and space at most:

$$S(\beta, n) \in O^*\left(\binom{n}{\beta n}\right).$$

Now, we run the main algorithm until the remaining graph has size βn or less; then a polynomial-time query in the storage table allows us to conclude. Thus, total running time and space is:

$$O^*\left(\max\left\{\binom{n}{\beta n}, 2^{2(1-\beta)n/5}\right\}\right).$$

This value is minimum for a β solution of the equation $2^{2(1-\beta)/5} = 1/(\beta^\beta(1-\beta)^{1-\beta})$ that leads to $T(n) \leqslant O^*(1.2937^n)$. □

4.2. MIN DOMINATING CLIQUE

The function $T(S, D, A, F)$ is now an integer function that returns the cardinality of a minimum dominating clique in G contained in set A and with no vertex from set D, if any, and ∞ otherwise. Once again, solving MIN DOMINATING CLIQUE is equivalent to finding $\min_{v\in V}\{T(\{v\}, \emptyset, N(v), V \setminus N[v])\}$. The following lemmata hold, the proof of the former (Lemma 3) being obvious.

Lemma 3. *Each vertex $j \in A$ such that $d_F(j) = 0$ can be discarded.*

Lemma 4. *If for all $j \in A$ $\bar{d}_A(j) = 0$ then the problem is solvable in time $O^*(1.2301^n)$ and polynomial space.*

Proof. Indeed, in this case A is a clique. The solution of MIN DOMINATING CLIQUE becomes the solution of a pure set covering problem where the set F corresponds to the universe U of elements and the set A corresponds to the collection S of the (nonempty) subsets of U and the aim is to determine a minimum cardinality sub-collection $S' \subseteq S$ which covers U. This set covering problem is known to be solvable to optimality in time $O^*(1.2301^{|A|+|F|})$ [12]. But, as $|A| + |F| \leqslant n$ this is not superior to $O^*(1.2301^n)$ time. □

Lemma 5. *If there exists $i \in F$ which is adjacent to at most 2 vertices in A, then we can branch in such a way that $T(n) \leqslant 2T(n-3)$.*

Proof. Indeed, if $i \in F$ is adjacent only to the vertex $j \in A$, then no branch occurs, j is included in S dominating i which is removed from F; alternatively, $i \in F$ is adjacent to vertices $j, k \in A$ and either j or k must be included in S to dominate i. Then, if $d_F(j) \geqslant 2$, either j is included in S and at least 3 vertices are removed, or j is discarded, k is included and i is dominated, i.e., 3 vertices are removed. If $d_F(j) = d_F(k) = 1$, $\bar{d}_A(j) \geqslant 1$ holds, or else k could be discarded without branching. But then, in both cases at least 3 vertices are fixed leading to $T(n) \leqslant 2T(n-3)$ corresponding to $O^*(1.26^n)$ time. □

We claim that we can solve MIN DOMINATING CLIQUE with running time $O^*(1.3248^n)$ by the following algorithm, called MINDC, that works as follows (as for Algorithm EDC, Case 5, which implies various branching rules, will be detailed in the proof of Proposition 3 that follows):

1. if $A = F = \emptyset$, then $T(S, D, A, F) = |S|$;
2. else, if $\exists i \in F$, such that $d_A(i) = 0$, then $T(S, D, A, F) = \infty$;
3. else, if $\forall j \in A$ $\bar{d}_A(j) = 0$, then solve the problem as a minimum set covering problem according to Lemma 4 (with complexity at most $O^*(1.2301^n)$);
4. else, if $\exists i \in F$, such that $2 \geqslant d_A(i) \geqslant 1$, then branch on the vertices adjacent to i according to Lemma 5 (with complexity at most $O^*(1.26^n)$);
5. else, if $\forall i \in F$ $d_A(i) \geqslant 3$, then select $j \in A$ such that $\bar{d}_A(j)$ is maximum and branch according to the following exhaustive cases:
   (a) $\bar{d}_A(j) \geqslant 1$ with vertex $j \in A$ adjacent to only one vertex $i \in F$ with $d_A(i) = 3$;
   (b) $\bar{d}_A(j) \geqslant 1$ with vertex $j \in A$ adjacent to only one vertex $i \in F$ with $d_A(i) \geqslant 4$;
   (c) $\bar{d}_A(j) = 1$ with vertex $j \in A$ non adjacent to vertex $k \in A$ and adjacent to exactly two vertices $h, i \in F$ with $d_A(i) \geqslant d_A(h) = 3$;

(d) $\bar{d}_A(j) = 1$ with vertex $j \in A$ non adjacent to vertex $k \in A$ and adjacent to exactly two vertices $h, i \in F$ with $d_A(i) \geqslant d_A(h) = 4$;
(e) $\bar{d}_A(j) = 1$ with vertex $j \in A$ non adjacent to vertex $k \in A$ and adjacent to exactly two vertices $h, i \in F$ with $d_A(i) \geqslant d_A(h) \geqslant 5$;
(f) $\bar{d}_A(j) = 1$ with vertex $j \in A$ non adjacent to vertex $k \in A$ and adjacent to at least three vertices $g, h, i \in F$;
(g) $\bar{d}_A(j) \geqslant 2$ with vertex $j \in A$ non adjacent to vertices $k, m \in A$ and adjacent to at least two vertices $h, i \in F$.

Proposition 3. *Algorithm MINDC computes a minimum-size dominating clique, if any, with running time $O^*(1.3248^n)$ and polynomial space, while, using memoization, it can run in time and space $O^*(1.2980^n)$.*

Proof. The correctness of the algorithm is straightforward. For the complexity we consider all subcases of case 5.

- If case 5a holds, let j, k, l be the three vertices $\in A$ adjacent to i. If j is selected, it must dominate i, hence k, l must be discarded. Alternatively, j is discarded inducing $d_A(i) = 2$ and the applicability of Lemma 5. Overall, we have $T(n) \leqslant T(n-4) + 2T(n-4) = 3T(n-4)$ inducing as complexity $O^*(1.3161^n)$.
- If case 5b holds, there are at least 4 vertices $j, k, l, m \in A$ adjacent to i. If j is selected, it must dominate i, hence k, l, m must be discarded. Alternatively, j is discarded. Hence, either 1 vertex or 5 vertices are fixed, i.e., we have $T(n) \leqslant T(n-1) + T(n-5)$ inducing as complexity $O^*(1.3248^n)$.

Note that in all the following items of the proof, $d_F(j) \geqslant 2$.

- If case 5c holds, then either j is selected, $k \in A$ non adjacent to j is discarded and h, i are dominated, or j is discarded inducing $d_A(h) = 2$ and the applicability of Lemma 5. Overall, we have $T(n) \leqslant T(n-4) + 2T(n-4) = 3T(n-4)$ inducing as complexity $O^*(1.3161^n)$.
- If case 5d holds, then j is non adjacent to $k \in A$ and is adjacent to $h, i \in F$ with $d_A(i) \geqslant d_A(h) = 4$; note also that, by the assumptions on j, $\bar{d}_A(k) = 1$. If $\exists l \in A$ with $\bar{d}_A(l) = 0$ which is adjacent to both h and i, then j is superseded by l and can be discarded. Also, if l is adjacent to just one among h and i, say, adjacent to h and non adjacent to i, then all other vertices adjacent to i must be discarded when selecting j or else j would again be superseded by l: but this induces a recurrence $T(n) \leqslant T(n-1) + T(n-6)$ with complexity $O^*(1.286^n)$. Similar considerations hold for vertex k. If k is adjacent to both h and i, then j is superseded by k and can be discarded. Else, if k is adjacent to just one among h and i, say, adjacent to h and non adjacent to i, then all other vertices adjacent to i must be discarded when selecting j or else j would be superseded by k: once again, this, induces a recurrence $T(n) \leqslant T(n-1) + T(n-6)$ with complexity $O^*(1.286^n)$. Alternatively, neither k nor any $l \in A$ with $\bar{d}_A(l) = 0$ are adjacent to both h and i. Let α, β and $\gamma \in A$ be the available vertices adjacent to h. Notice that $\bar{d}_A(\alpha) = \bar{d}_A(\beta) = \bar{d}_A(\gamma) = 1$ as $\bar{d}_A(j) = 1$, j being the vertex with maximum anti-degree within set A and must have anti-degree within set A strictly greater than 0 and they are adjacent to h. Then, there are at least 2 edges between vertices α, β and γ, say, edges $(\alpha\beta)$ and $(\alpha\gamma)$. Consequently, a branch on the vertices dominating h holds, where either j is selected or j is discarded and α is selected or j and α are discarded and β is selected or j, α and β are discarded and γ is selected. We can assume $d_F(\alpha) \geqslant 2$; otherwise, one can apply cases 5a or 5b with α instead of j. All these induce a recurrence $T(n) \leqslant T(n-4) + T(n-5) + 2T(n-6)$ with complexity $O^*(1.3086^n)$.
- If case 5e holds, then j is non adjacent to $k \in A$ and is adjacent to $h, i \in F$ with $d_A(i) \geqslant d_A(h) \geqslant 5$. Similarly to subcase 5d, if $\exists l \in A$ with $\bar{d}_A(l) = 0$ which is adjacent to either h or i or both, or if k is adjacent to either h or i or both, then a recurrence $T(n) \leqslant T(n-1) + T(n-7)$ would hold (now $d_A(i) \geqslant d_A(h) \geqslant 5$) with complexity $O^*(1.2555^n)$. Alternatively, a branch on j holds where, if j is selected, then k is discarded, h and i are dominated and either all other vertices $\in A$ and adjacent to h are discarded, or all other vertices $\in A$ and adjacent to i are discarded. This induces a recurrence $T(n) \leqslant T(n-1) + 2T(n-8)$ with complexity $O^*(1.307^n)$.
- If case 5f holds, vertex j is non adjacent to vertex $k \in A$ and is adjacent to at least 3 vertices $g, h, i \in F$. If j is selected, k is discarded and g, h, i are dominated. Alternatively, j is discarded. Hence, either 1 vertex or 5 vertices are fixed, i.e., we have $T(n) \leqslant T(n-1) + T(n-5)$ inducing as complexity $O^*(1.3248^n)$.
- If case 5g holds, vertex j is non adjacent at least to vertices $k, m \in A$ and is adjacent at least to vertices $h, i \in F$. If j is selected, k, m are discarded and h, i are dominated. Alternatively, j is discarded. Hence, either 1 vertex or 5 vertices are fixed, i.e., we have $T(n) \leqslant T(n-1) + T(n-5)$ inducing as complexity $O^*(1.3248^n)$.

The overall worst-case complexity is then $O^*(1.3248^n)$.

For trading space for time, one can do a previously. First, one solves the problem in a bottom up fashion on every subgraph of size at most γn and stores the results in a table, with time and space at most:

$$S(\gamma, n) \in O^*\left(\binom{n}{\gamma n}\right).$$

Now, one runs the main algorithm until the remaining graph has size γn or less; then a polynomial-time query in the storage table allows one to conclude. Thus, total running time and space is:

$$O^*\left(\max\left\{\binom{n}{\gamma n}, 1.3248^{(1-\gamma)n}\right\}\right)$$

This value is minimum for a γ solution of the equation $1.3248^{1-\gamma} = 1/(\gamma^\gamma(1-\gamma)^{1-\gamma})$, that leads to $T(n) = O^*(1.2980^n)$. □

5. Dominating cliques in dense and sparse graphs

5.1. Graphs of fixed maximum or minimum degree

Notice that if a graph has maximum degree Δ, all the maximal cliques can be computed with running time $O^*(3^{\Delta/3})$ and all the cliques with running time $O^*(2^{\Delta})$. In particular, dominating clique problems are polynomial if $\Delta = O(\log n)$ and subexponential if $\Delta = \Omega(\log n)$, but remains $o(n)$.

In the case of high minimum degree δ, this does not remain true. Indeed, MAX CLIQUE easily reduces to MAX DOMINATING CLIQUE by adding to the graph instance G a new vertex adjacent to any vertex in G, and MAX CLIQUE is well known to be **NP**-hard even in graphs of minimum degree $n-4$ (MAX INDEPENDENT SET being **NP**-hard in graphs of maximum degree 3 [14]). Then, even in graphs with minimum degree $\delta \geqslant n - \overline{\delta}$ for some constant $\overline{\delta}$ MAX DOMINATING CLIQUE is not polynomial (if **P**$\neq$ **NP**). However, there are some interesting results.

For $v \in V$, let us define $f(v) = 1 - |N(v)|/|V|$. Then, for any $v \in V$, $nf(v) \leqslant n - \delta$. Thanks to Lemma 2, if there exists a dominating clique, then there exists a dominating clique of size at most $nf(v)$. Thus, EXISTING, and MIN DOMINATING CLIQUES can be computed with running time $O^*\left(\binom{n}{n-\delta}\right)$ for $\delta > n/2$. In particular, EXISTING, and MIN DOMINATING CLIQUES are polynomial if $n - \delta$ is finite and subexponential if $n - \delta = o(n)$.

We now exhibit Algorithm MDC1 that solves MAX DOMINATING CLIQUE in any graph, but it is interesting only for dense graphs:

1. for every $v \in V$, form the collection $\mathcal{S}(v)$ of every subset of $N[v]$ of size at most $nf(v)$ that is a dominating clique;
2. for every $S \in \mathcal{S}(v)$, compute a maximum clique in $G[\bigcap_{s\in S} N[s]]$;
3. return the maximum-size clique among those computed in Step 2.

Proposition 4. *Algorithm MDC1 solves* MAX DOMINATING CLIQUE.

Proof. Let K^* be an optimal solution for MAX DOMINATING CLIQUE. According to Lemma 2, for any $v \in K^*$, there exists $K \subset K^*$ such that K is a dominating clique and $|K| \leqslant nf(v) \leqslant n-\delta$. Thus, K belongs to $\mathcal{S}(u)$, for some u. Let K' be the maximum clique in $G[\bigcap_{s\in K} N[s]]$. According to Lemma 1, K' is a dominating clique, while $K^* \subseteq \bigcap_{s\in K} N[s]$. Then, by maximality, $|K'| = |K^*|$. So, Algorithm MDC1 computes a maximum-size dominating clique. □

For $\delta \geqslant n/2$, the size of the collection computed at step 1 is at most $n^2\binom{n}{n-\delta}$; thus, Algorithm MDC1 has running time $O^*\left(c^n\binom{n}{n-\delta}\right)$ if the algorithm used to solve MAX CLIQUE has complexity $O^*(c^n)$. In particular, MAX DOMINATING CLIQUE can be computed with the same exponential bound on running time as MAX CLIQUE in graphs such that $n - \delta = o(n)$; this bound cannot be improved (thanks to the reduction from MAX CLIQUE mentioned above).

Note that the best worst-case complexity bound for MAX CLIQUE is, to our knowledge, the (exponential space) $O^*(1.1889^n)$ time bound claimed by [24] in his technical report, or the (polynomial space) $O^*(1.2125^n)$ algorithm by [5].

5.2. Graphs of small average degree

More generally, the density of a graph can be defined as $D(G) = 2m/(n(n-1)) = d/(n-1)$ where d is the average degree of the graph. It can be easily seen that, if $D(G) = o(1)$, the size of a maximum clique is $o(n)$ and then we can enumerate all cliques in subexponential time.

We now focus ourselves on the case where $D(G) \notin o(1)$ but is bounded above by some constant $\lambda \in [0, 1]$. In this case, average degree is at most $\lambda(n-1)$, hence:

$$m \leqslant \frac{\lambda n(n-1)}{2} \leqslant \frac{\lambda n^2}{2}. \tag{1}$$

By exhaustive enumeration of all the subsets of size at most $\sqrt{D(G)}n$, we trivially find if there is a dominating clique (and return the minimal and the maximal one) with running time $O^*\left(\binom{n}{\sqrt{\lambda}n}\right)$. This is only interesting for rather small values of λ; for example, if $\sqrt{\lambda} = 1/20$, running time is $O^*(1.22^n)$.

As far as EXISTING DOMINATING CLIQUE and MAX DOMINATING CLIQUE are concerned (not MIN DOMINATING CLIQUE), we can greatly improve this result. Notice that, for any dominating clique K we have the following inequality:

$$m \geqslant \frac{|K|(|K|-1)}{2} + \sum_{v\in K} d_{V\setminus K}(v).$$

Assume first that there exists a vertex $v \in K$ such that $d_{V\setminus K}(v) < \mu n + 1/2$, for a given μ whose value will be fixed later. Then $n(1 - f(v)) < \mu n + 1/2 + |K|$.

In [6] it is established that in a graph of size n, the cliques of size k or less (where $n/k \geqslant 3$) can be enumerated in time $O^*((n/k)^k)$. Thus, we can enumerate all maximal cliques containing v with running time:

$$T(n) = O^*\left(\left(\frac{\mu n + \frac{1}{2} + |K|}{|K|}\right)^{|K|}\right) = O^*\left(2^{|K|\log_2(1+\mu n/|K|+1/2|K|)}\right).$$

Table 3
Running time of our algorithm vs. running time of exhaustive enumeration of all the subsets of size at most $\sqrt{D(G)}n$, for some values of λ.

$\sqrt{\lambda}$	1/4	1/6	1/8	1/10	1/20	1/50
Optimal μ	0.326	0.248	0.200	0.169	0.097	0.045
Running time of our algorithm	1.233^n	1.164^n	1.127^n	1.104^n	1.056^n	1.024^n
Running time $O^*\left(\binom{n}{\sqrt{\lambda}n}\right)$	1.755^n	1.570^n	1.458^n	1.385^n	1.220^n	1.104^n

Since this function is increasing with K and, according to (1), $|K| \leqslant \sqrt{2m} + 1 \leqslant \sqrt{\lambda}n + 1$, this leads to:

$$T(n) = O^*\left(2^{\sqrt{\lambda}n \log_2(1+\mu/\sqrt{\lambda})}\right). \tag{2}$$

On the other hand, if for any $v \in K$, $d_{V\setminus K}(v) \geqslant \mu n + 1/2$, using also (1), we get:

$$m \geqslant \frac{|K|(|K|-1)}{2} + |K|\left(\mu n + \frac{1}{2}\right) = \frac{|K|^2}{2} + |K|\mu n \Longrightarrow |K| \leqslant \left(\sqrt{\lambda+\mu^2} - \mu\right) n.$$

In this case, running time for enumerating all small subsets containing v is bounded above by

$$T(n) = O^*\left(\left(\frac{n}{\left(\sqrt{\lambda+\mu^2}-\mu\right)n}\right)^{\left(\sqrt{\lambda+\mu^2}-\mu\right)n}\right) = O^*\left(2^{\left(\sqrt{\lambda+\mu^2}-\mu\right)n\left|\log_2\left(\sqrt{\lambda+\mu^2}-\mu\right)\right|}\right). \tag{3}$$

The running time given by (2) is increasing with μ while the one given by (3) is decreasing. So, the best value for μ is given when both are equal,[2] i.e. when

$$\left(\sqrt{\lambda+\mu^2}-\mu\right)\cdot\left|\log_2\left(\sqrt{\lambda+\mu^2}-\mu\right)\right| = \sqrt{\lambda}\log_2\left(1+\frac{\mu}{\sqrt{\lambda}}\right).$$

In Table 3, bounds on running time of the above method vs. running time of exhaustive search (both depending on parameter λ) are given.

5.3. Graphs of high average degree

In this section, we deal with graphs of density $D(G) \geqslant 3/4$. As previously, it is easy to see that in such graphs MAX DOMINATING CLIQUE is harder than MAX CLIQUE.

Let us define Algorithms mDC and MDC2 for MIN DOMINATING CLIQUE and MAX DOMINATING CLIQUE, respectively. The former, mDC, works as follows:

> fix $\epsilon = \sqrt{1-D(G)}$, compute any subset of size at most ϵn and return a smallest one that is a dominating clique if any.

On the other hand, Algorithm MDC2 for MAX DOMINATING CLIQUE works as follows:

1. fix $\epsilon = \sqrt{1-D(G)}$ and compute any subset of size at most ϵn; let K_0 be a largest one that is a dominating clique, if any;
2. for every $v \in V$ such that $nf(v) \leqslant \epsilon n$, form the collection $\mathcal{S}(v) \subseteq N[v]$ of every subset of size at most $nf(v)$ that is a dominating clique;
3. for every $S \in \mathcal{S}(v)$, compute a maximum clique in $G[\bigcap_{s\in S} N[s]]$;
4. let K_1 be a clique of maximum size among those computed in Step 3; output $\max\{K_0, K_1\}$.

Obviously, if there exists no dominating clique in the graph, both algorithms return nothing. We now prove that, if some dominating clique exists, Algorithm mDC computes a minimum dominating clique, while Algorithm MDC2 computes a maximum dominating clique.

Fix K^* a maximum dominating clique. If $|K^*| \leqslant \epsilon n$, it is clear that both algorithms work correctly. Assume $|K^*| > \epsilon n$ and suppose that, for any $v \in K^*, f(v) \geqslant \epsilon$. Then:

$$\frac{n(n-1)}{2} - m \geqslant \frac{1}{2}\sum_{v\in K^*} nf(v) \geqslant \frac{\epsilon^2}{2}n^2 > \frac{1-D(G)}{2}n(n-1).$$

Plugging $m = D(G)n(n-1)/2$ in the previous inequality leads to a contradiction.

[2] Under the condition that $n/k \geqslant 3$, i.e. $\sqrt{\lambda+\mu^2} - \mu \leqslant 1/3$, which is verified in the sequel.

Consequently, there exists a vertex $v \in K^*$ with $nf(v) < \epsilon n$. Since there exists a dominating clique of size at most $nf(v)$, according to Lemma 2, Algorithm mDC returns a minimum dominating clique K_0. Furthermore, K^* has the same size as a maximum clique in $G[\bigcap_{s\in K_0} N[s]]$, which means that Algorithm MDC2 returns a maximum dominating clique.

To conclude, notice that $D(G) = 1 - o(1)$ implies $\epsilon = o(1)$, and then

$$\binom{n}{\epsilon n} = O^*\left(\left(\frac{1}{\epsilon^\epsilon(1-\epsilon)^{1-\epsilon}}\right)^n\right) = 2^{o(n)}.$$

So, in this case Algorithm mDC has subexponential running time, while the complexity of Algorithm MDC2 is the running time of the MAX CLIQUE-algorithm called in Step 3, with a subexponential multiplicative factor. Again, no improvement in the exponential basis of the running time seems possible thanks to the reduction from MAX CLIQUE to MAX DOMINATING CLIQUE mentioned above.

6. Some words on moderately exponential approximation for the dominating clique

Since any approximation algorithm for MIN DOMINATING CLIQUE or MAX DOMINATING CLIQUE solves EXISTING DOMINATING CLIQUE that is **NP**-complete, it is impossible to devise any polynomial approximation algorithm for MIN and MAX DOMINATING CLIQUES. Thus, it seems interesting to see if it is possible to compute solutions for the two optimization problems that guarantee a good approximation ratio with moderately exponential running time; interesting running times of approximation algorithms lie between the best known complexity for solving EXISTING DOMINATING CLIQUE ($O^*(1.2628)$ with our algorithm) and the best known complexity for solving the corresponding optimization problem.

We propose in what follows two algorithms mMOD and MMOD that do this for MIN DOMINATING CLIQUE and MAX DOMINATING CLIQUE, respectively. Obviously, in what follows we assume that we handle graphs admitting dominating cliques (otherwise our algorithm detects that no dominating clique exists).

Algorithm mMOD(ρ) (i.e., parameterized by the ratio ρ that one wishes to attain) for MIN DOMINATING CLIQUE works as follows:

- run algorithm EDC and let K_0 be the dominating clique computed;
- compute all the subsets of V whose size is at most n/ρ; let K_1 be a dominating clique of minimum size among them; if none is found, then set $K_1 = V$;
- return argmin$\{|K_0|, |K_1|\}$.

Proposition 5. *If $G = (V, E)$ has a dominating clique then:*

1. *for any ρ, $2 \leqslant \rho \leqslant 16.03$, it is possible to compute a ρ-approximation to* MIN DOMINATING CLIQUE *with polynomial space and running time $O^*\left(\binom{n}{n/\rho}\right)$; this is faster than the exact (polynomial space) algorithm for $\rho \geqslant 12.35$;*
2. *for any ρ, $2 \leqslant \rho \leqslant 17.45$, it is possible to compute a ρ-approximation to* MIN DOMINATING CLIQUE *with running time and space $O^*\left(\binom{n}{n/\rho}\right)$; this is faster than the exact (exponential space) algorithm for $\rho \geqslant 13.74$.*

Proof. For item 1, observe first that Algorithm mMOD has running time (for $2 \leqslant \rho \leqslant 16.03$):

$$O^*\left(1.2628^n + \binom{n}{n/\rho}\right) = O^*\left(\binom{n}{n/\rho}\right).$$

Now, let K_m^* be a minimum dominating clique of G. If $|K_m^*| \leqslant n/\rho$, then $|K_1| = |K_m^*|$ and mMOD is optimal. Otherwise:

$$\frac{|K_0|}{|K_m^*|} \leqslant \frac{n}{\frac{n}{\rho}} = \rho.$$

Some simple algebra fixes ρ as claimed. Item 2 is proved in a similar way by using memoization. □

Unfortunately, this strategy does not work as far as MAX DOMINATING CLIQUE is concerned. Indeed, consider a graph on $n + 3$ vertices, consisting of a clique K of size $n/3$, a triangle T and an independent set S of size $2n/3$ (for some n multiple of 3). Split the vertices of K and S into three equal parts K_1, K_2, K_3 and S_1, S_2, S_3, respectively, and denote by t_1, t_2 and t_3, the vertices of T. For $i = 1, 2, 3$, draw the edges $(t_i, x_{i_j}), x_{i_j} \in K_i, j = 1, \ldots, n/9$ (in other words link t_i to all the vertices of K_i) as well as the edges $(t_i, y_{i_j}), y_{i_j} \in S_i, j = 1, \ldots, 2n/9$ (in other words link t_i to all the vertices of S_i). Finally, for $i = 1, 2, 3$, link each vertex of K_i to 2 ("private") vertices of S_i. It is easy to see that in the so built graph K and T are the only dominating cliques. There, algorithm EDC could return T and taking it as solution for MAX DOMINATING CLIQUE, this is a bad solution.

However, it is possible to get some approximation results if we observe that *algorithm EDC is able not only to find a dominating clique, but also, given a subset S, to find a dominating clique containing S.*

Based upon this remark we devise Algorithm MMOD(ρ) for MAX DOMINATING CLIQUE that works as follows:

- compute all the subsets K of V whose size is at most n/ρ; if K is a clique, then find:

$$T\left(K, N[K] \setminus \bigcap_{v\in K} N[v], \bigcap_{v\in K} N(v), V \setminus N[K]\right)$$

  according to algorithm EDC (where $N[K] = \cup_{v\in K} N[v]$);
- return the largest dominating clique found, denoted by K.

Proposition 6. *If $G = (V, E)$ has a dominating clique, then:*

1. *for any $\rho \geqslant 2$ it is possible to compute a ρ-approximation to* MAX DOMINATING CLIQUE *with polynomial space and running time:*

$$O^*\left(\binom{n}{n/\rho} 1.2628^{n(1-1/\rho)}\right)$$

   and is faster than the exact (polynomial space) algorithm for $\rho \geqslant 128$;
2. *for any $\rho \geqslant 2$ it is possible to compute a ρ-approximation to* MAX DOMINATING CLIQUE *with running time and space:*

$$O^*\left(\binom{n}{n/\rho} 1.2453^{n(1-1/\rho)}\right)$$

   and is faster than the exact (exponential space) algorithm for $\rho \geqslant 153$.

Proof. For item 1, observe first that Algorithm MMOD has running time:

$$O^*\left(1.2628^{|V\setminus N[K]|}\binom{n}{|K|}\right) = O^*\left(1.2628^{n(1-1/\rho)}\binom{n}{n/\rho}\right)$$

Now, let K_M^* be a maximum dominating clique of G. If $|K_M^*| \leqslant n/\rho$, then MMOD computes an optimal solution. Otherwise, it computes at least a dominating clique containing a subclique of size n/ρ thus guaranteeing $|K_M^*|/|K| \leqslant n/(n/\rho) = \rho$, as claimed.

Item 2 is proved similarly just using memoization. □

7. Conclusion

We have designed some simple exact algorithms for dominating clique problems with worst case complexity improving upon previous results. Next, we have handled these problems in natural graph families, namely families of dense and sparse graphs, further improving the general worst-case time bounds for these families. Finally, we have proposed moderately exponential approximation algorithms for MIN DOMINATING CLIQUE and MAX DOMINATING CLIQUE.

Let us note that moderately exponential approximation is meaningful for these two problems since, as already mentioned, question of polynomially approximating them is senseless. More generally, we think that moderately exponential approximation is an interesting research program for breaking barriers of polynomial approximation. Moreover, we feel that this the best one can do in approximation. To make a long story short, an interesting issue for the approximation of **NP**-hard problems is to approximate them by subexponential algorithms guaranteeing approximation ratios unachievable in polynomial time. We feel that such an issue is very difficult, not to say impossible, for many problems unless the exponential time hypothesis (claiming that there exists an $\epsilon > 0$ such that no algorithm solves 3 SAT in time $2^{\epsilon n}$, where n is the number of variables [15]) fails. It is true that, for several problems that are hard to approximate in polynomial time (like MAX INDEPENDENT SET, MIN COLORING, ...), subexponential time can be easily reached for ratios depending on the input size. But approximation of either MIN COLORING or MAX INDEPENDENT SET within constant ratios in subexponential time seems unlikely under the exponential time hypothesis [11].

Finally, another analogous issue is the FPT approximability of hard problems. Here the objective is to devise approximation algorithms running in FPT time that either returns a solution of value rk, or returns "no", asserting in this latter case that there is no solution of value k. But this issue receives negative answers for the dominating clique. Indeed, FPT approximation for k-dominating cliques (in either one of the optimization versions) would solve the existing dominating clique in FPT time, that is impossible given that this latter problem is **W[2]**-hard.

Acknowledgment

The very pertinent and useful comments of an anonymous referee are gratefully acknowledged.

References

[1] A. Björklund, T. Husfeldt, M. Koivisto, Set partitioning via inclusion-exclusion, SIAM J. Comput. 39 (2) (2009) 546–563.
[2] N. Bourgeois, B. Escoffier, V.Th. Paschos, Efficient approximation of MIN COLORING by moderately exponential algorithms, Inform. Process. Lett. 109 (16) (2009) 950–954.
[3] N. Bourgeois, B. Escoffier, V.Th. Paschos, Efficient approximation of MIN SET COVER by moderately exponential algorithms, Theoret. Comput. Sci. 410 (21–23) (2009) 2184–2195.
[4] N. Bourgeois, B. Escoffier, V.Th. Paschos, Approximation of MAX INDEPENDENT SET, MIN VERTEX COVER and related problems by moderately exponential algorithms, Discrete Appl. Math. 159 (17) (2011) 1954–1970.
[5] N. Bourgeois, B. Escoffier, V.Th. Paschos, J.M.M. van Rooij, Fast algorithms for MAX INDEPENDENT SET, Algorithmica 62 (1–2) (2012) 382–415.
[6] J.M. Byskov, Enumerating maximal independent sets with applications to graph colouring, Oper. Res. Lett. 32 (6) (2004) 547–556.
[7] D.G. Corneil, Y. Perl, Clustering and domination in perfect graphs, Discrete Appl. Math. 9 (1984) 27–39.
[8] M.B. Cozzens, L.L. Kelleher, Dominating cliques in graphs, Discrete Appl. Math. 86 (1990) 101–116.
[9] M. Cygan, L. Kowalik, M. Wykurz, Exponential-time approximation of weighted set cover, Inform. Process. Lett. 109 (16) (2009) 957–961.
[10] M. Cygan, M. Pilipczuk, Exact and approximate bandwidth, Theoret. Comput. Sci. 411 (40–42) (2010) 3701–3713.
[11] B. Escoffier, E.J. Kim, V.Th. Paschos, Subexponential and FPT-time inapproximability of independent set and related problems, Cahier du LAMSADE 321, LAMSADE, Université Paris-Dauphine, May 2012.
[12] F.V. Fomin, F. Grandoni, D. Kratsch, A measure & conquer approach for the analysis of exact algorithms, J. Assoc. Comput. Mach. 56 (5) (2009) 1–32.
[13] M. Fürer, S. Gaspers, S.P. Kasiviswanathan, An exponential time 2-approximation algorithm for bandwidth, in: Proc. International Workshop on Parameterized and Exact Computation, IWPEC'09, in: Lecture Notes in Computer Science, vol. 5917, Springer, 2009, pp. 173–184.
[14] M.R. Garey, D.S. Johnson, Computers and Intractability. A Guide to the Theory of NP-Completeness, W.H. Freeman, San Francisco, 1979.
[15] R. Impagliazzo, R. Paturi, F. Zane, Which problems have strongly exponential complexity? J. Comput. System Sci. 63 (4) (2001) 512–530.
[16] D.S. Johnson, M. Yannakakis, C.H. Papadimitriou, On generating all maximal independent sets, Inform. Process. Lett. 27 (1988) 119–123.
[17] L.L. Kelleher, Domination in graphs and its application to social network theory, Ph.D. Thesis, Northeastern University, 1985.
[18] L.L. Kelleher, M.B. Cozzens, Dominating sets in social network graphs, Math. Social Sci. 16 (3) (1988) 267–279.
[19] D. Kratsch, Algorithms, in: T. Haynes, S. Hedetniemi, P. Slater (Eds.), Domination in Graphs: Advanced Topics, Marcel Dekker, 1988, pp. 191–231.
[20] D. Kratsch, M. Liedloff, An exact algorithm for the minimum dominating clique problem, Theoret. Comput. Sci. 385 (1–3) (2007) 226–240.
[21] J.W. Moon, L. Moser, On cliques in graphs, Israel J. Math. 3 (1965) 23–28.
[22] V. Raman, S. Saurabh, S. Sikdar, Efficient exact algorithms through enumerating maximal independent sets and other techniques, Theory Comput. Syst. 41 (3) (2007) 563–587.
[23] J.M. Robson, Algorithms for maximum independent sets, J. Algorithms 7 (3) (1986) 425–440.
[24] J.M. Robson, Finding a maximum independent set in time $O(2^{n/4})$, Technical Report 1251-01, LaBRI, Université de Bordeaux I, 2001.